Chapter 7

Analysis and Modeling with REST Services and Microservices

This chapter provides a detailed step-by-step process for modeling REST service candidates.

7.1 REST Service Modeling Process

The incorporation of resources and uniform contract features adds new dimensions to service modeling. When we are aware that a given service candidate is being modeled specifically for a REST implementation, we can take these considerations into account by extending the service modeling process to include steps to better shape the service candidate as a basis for a REST service contract.

The REST service modeling process shown in Figure 7.1 provides a generic set of steps and considerations tailored for modeling REST services. This chapter describes each process step and is further supplemented with case study examples.

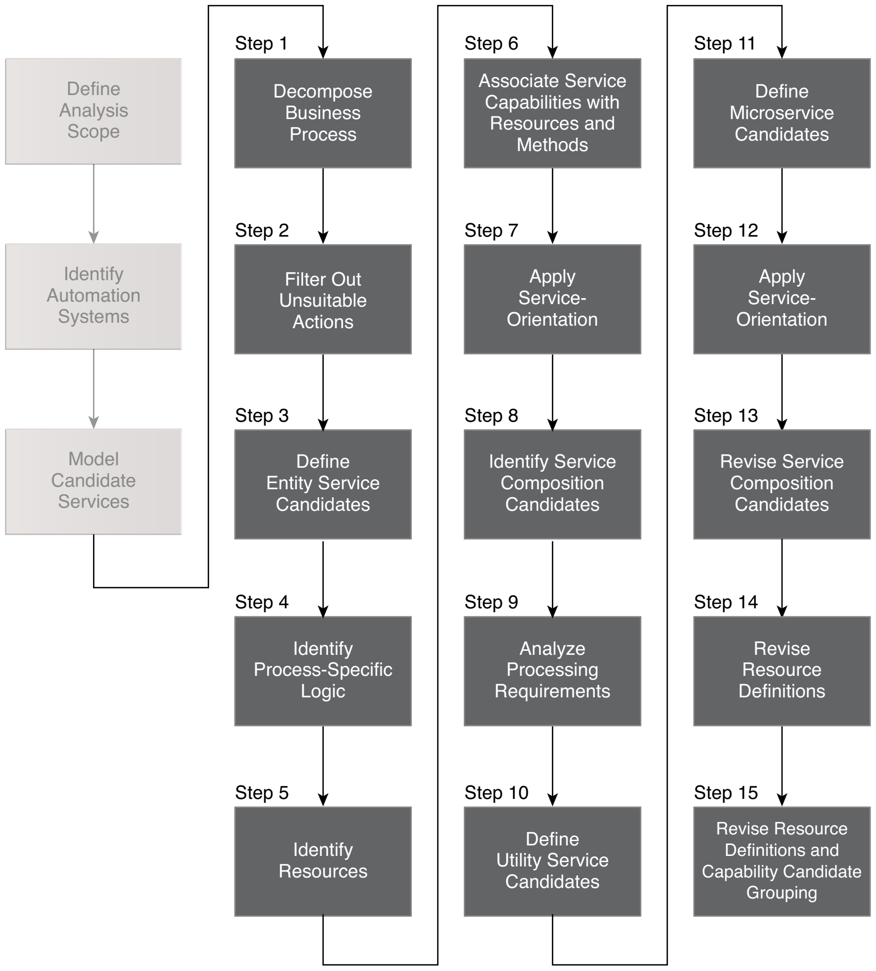
Define Analysis Scope
Identify Automation Systems
Model Candidate Services
Step 1
Decompose Business Process
Step 2
Filter Out Unsuitable Actions
Step 3
Define Entity Service Candidates
Step 4
Identify Process-Specific Logic
Step 5
Identify Resources
Step 6
Associate Service Capabilities with Resources and Methods
Step 7
Apply Service-Orientation
Step 8
Identify Service Composition Candidates
Step 9
Analyze Processing Requirements
Step 10
Define Utility Service Candidates
Step 11
Define Microservice Candidates
Step 12
Apply Service-Orientation
Step 13
Revise Service Composition Candidates
Step 14
Revise Resource Definitions
Step 15
Revise Resource Definitions and Capability Candidate Grouping

Figure 7.1
A sample service modeling process for REST services.

CASE STUDY EXAMPLE

MUA architects are dedicated to adopting SOA and applying service-orientation as part of a key strategy to consolidate systems and data. They decide to focus on entity services that track the information assets of the various campuses. This initial set of services is to be deployed on the main campus first, so that IT staff can monitor maintenance requirements. Individual campuses are then to build solutions based on the same centralized service inventory. Solutions that introduce new task services will be allocated to virtual machines in the main campus to allow them to be moved to independent hardware and onto dedicated server farms, if the need arises in the future.

Existing MUA charter agreements with partner schools explicitly refer to the need to acknowledge individual academic achievements. This makes the correct conferral of awards important to the reputation of MUA and its elite students.

MUA assembles a service modeling team comprised of SOA architects, SOA analysts, and business analysts. The team begins with a REST service modeling process for the Student Achievement Award Conferral business process. As detailed in Figure 7.2, this business process logic represents the procedures followed for the assessment, conference, and rejection of individual achievement award applications submitted by students. An application that is approved results in the conferral of the achievement award and a notification of the conferral to the student. An application that is rejected results in a notification of the rejection to the student.

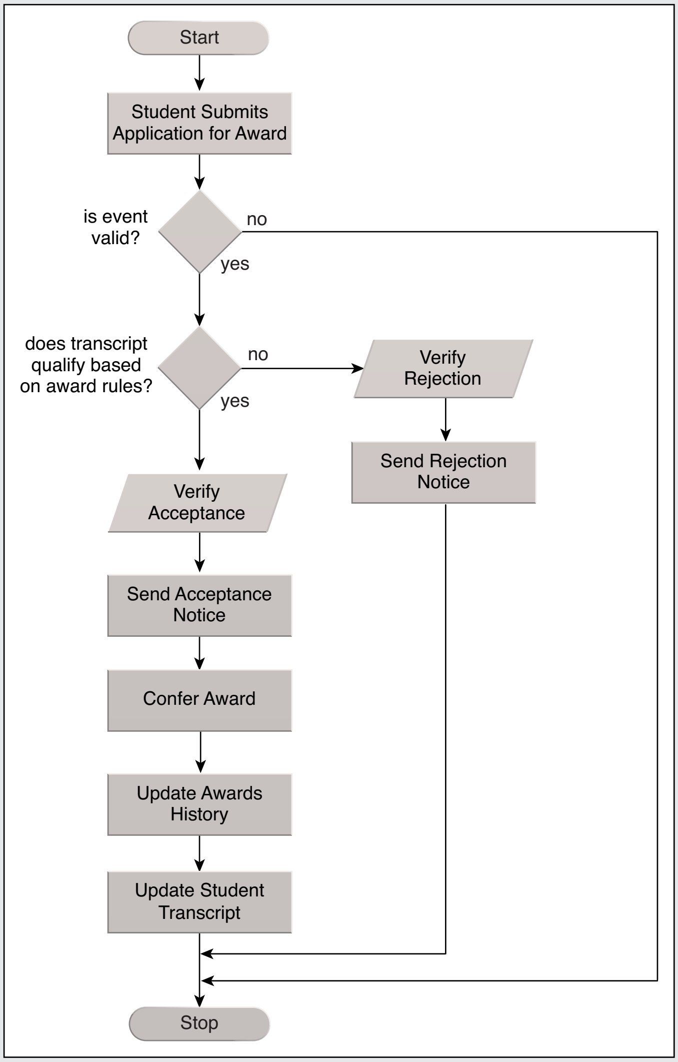

Figure 7.2
The Student Award Conferral business process.

Step 1: Decompose Business Process (into Granular Actions)

Let’s take the documented business process and break it down into a series of granular process steps. This requires further analysis of the process logic, during which we attempt to decompose the business process into a set of individual granular actions.

CASE STUDY EXAMPLE

The original Student Award Conferral business process is broken down into the following granular actions:

- Initiate Conferral Application
- Get Event Details
- Verify Event Details
- If Event is Invalid or Ineligible for Award, End Process
- Get Award Details
- Get Student Transcript
- Verify Student Transcript Qualifies for Award Based on Award Conferral Rules
- If Student Transcript Does Not Qualify, Initiate Rejection
- Manually Verify Rejection
- Send Rejection Notice
- Manually Verify Acceptance
- Send Acceptance Notice
- Confer Award
- Record Award Conferral in Student Transcript
- Record Award Conferral in Awards Database
- Print Hard Copy of Award Conferral Record
- File Hard Copy of Award Conferral Record

Step 2: Filter Out Unsuitable Actions

Not all business process logic is suitable for automation and/or encapsulation by a service. This step requires us to single out any of the granular actions identified in Step 1 that do not appear to be suitable for subsequent REST service modeling steps. Examples include manual process steps that need to be performed by humans and business automation logic being carried out by legacy systems that cannot be wrapped by a service.

CASE STUDY EXAMPLE

After assessing each of the decomposed actions, a subset is identified as being unsuitable for automation or unsuitable for service encapsulation, as indicated by the crossed-out items.

- Initiate Conferral Application
- Get Event Details
- Verify Event Details
- If Event is Invalid or Ineligible for Award, End Process
- Get Award Details
- Get Student Transcript
- Verify Student Transcript Qualifies for Award Based on Award Conferral Rules
- If Student Transcript Does Not Qualify, Initiate Rejection
- ~~Manually Verify Rejection~~
- Send Rejection Notice
- ~~Manually Verify Acceptance~~
- Send Acceptance Notice
- ~~Confer Award~~
- Record Award Conferral in Student Transcript
- Record Award Conferral in Awards Database
- Print Hard Copy of Award Conferral Record
- ~~File Hard Copy of Award Conferral Record~~

Step 3: Define Entity Service Candidates

By filtering out unsuitable actions during Step 2, we are left with only those actions relevant to our REST service modeling effort.

A primary objective of service-orientation is to carry out a separation of concerns whereby agnostic logic is cleanly partitioned from non-agnostic logic. By reviewing the actions that have been identified so far, we can begin to further separate those that have an evident level of reuse potential. This essentially provides us with a preliminary set of agnostic service capability candidates.

We then determine how these service capability candidates should be grouped to form the basis of functional service boundaries.

Common factors we can take into account include:

- Which service capability candidates defined so far are closely related to each other?
- Are identified service capability candidates business-centric or utility-centric?
- What types of functional service contexts are suitable, given the overarching business context of the service inventory?

The first consideration on the list requires us to group capability candidates based on common functional contexts. The second item pertains to the organization of service candidates within logical service layers based on service models. Due to the business-centric level of documentation that typically goes into the authoring of business process models and specifications and associated workflows, the emphasis during this step will naturally be more on the definition of entity service candidates. The upcoming Define Utility Service Candidates step is dedicated to developing the utility service layer.

The third item on the preceding list of factors relates to how we may choose to establish functional service boundaries not only in relation to the current business process we are decomposing, but also in relation to the overall nature of the service inventory. This broader consideration helps us determine whether there are generic functional contexts we can define that will be useful for the automation of multiple business processes.

SOA PATTERNS

Both the previously referenced Logic Centralization [348] and Service Normalization [361] patterns play a key role during this step to ensure we keep agnostic service candidates aligned to each other, without allowing functional overlap.

CASE STUDY EXAMPLE

By analyzing the remaining actions from Step 2, the MUA service modeling team identifies and categorizes those actions considered agnostic. Those that are classified as non-agnostic are in bold:

- **Initiate Conferral Application**
- Get Event Details
- **Verify Event Details**
- **If Event is Invalid or Ineligible for Award, Cancel Process**
- Get Award Details
- Get Student Transcript
- **Verify Student Transcript Qualifies for Award Based on Award Conferral Rules**
- **If Student Transcript Does Not Qualify, Initiate Rejection**
- Send Rejection Notice
- Send Acceptance Notice
- Record Award Conferral in Student Transcript
- Record Award Conferral in Awards Database
- Print Hard Copy of Award Conferral Record

Agnostic actions are classified as preliminary service capability candidates and are grouped accordingly into service candidates, as follows.

Event Service Candidate

The original Get Event Details action is positioned as a Get Details service capability candidate as part of an entity service candidate named Event (Figure 7.3).

Note that it was determined that the Verify Event Details action was not agnostic because it carried out logic specific to the Student Award Conferral process.

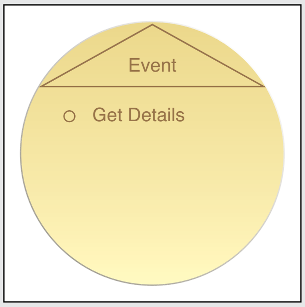

Figure 7.3
The Event service candidate with one service capability candidate.

Award Service Candidate

As a central part of this business process, the Award business entity becomes the basis of an Award entity service candidate (Figure 7.4).

The Get Award Details action establishes a Get Details service capability candidate. The Record Award Conferral in Awards Database action is split into two service capability candidates:

- Confer
- Update History

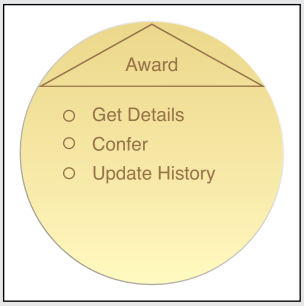

Figure 7.4
The Award service candidate with three service capability candidates, including two that are based on the same action.

The Confer capability is required to officially issue an award for an event, which requires updates in the internal MUA Awards database, as well as an update to an external National Academic Recognition System shared by schools throughout the U.S.

Furthermore, based on the award conferral policies, this service capability is required to issue a conferral notification and forward the award conferral record information to be printed in hard copy format. This relates to the following three actions:

- Send Rejection Notice
- Send Acceptance Notice
- Print Hard Copy of Award Conferral Record

The MUA team considers including this logic within the Award entity service, but then decides that the Confer service capability will instead invoke corresponding utility services to perform these functions automatically, upon each conferral.

The Update History capability will issue a further update of student and event details within a separate part of the internal Awards database. It is deemed necessary to keep the capabilities separate because the Update History capability can be used independently and for different purposes than the Confer capability.

Student Service Candidate

The need for a Student entity service within a school is self-evident. This service will eventually provide a wide range of student-related functions. In support of the Student Award Conferral business process specifically, the Get Student Transcript and Record Award Conferral in Student Transcript actions are positioned as individual service capability candidates named Get Transcript and Update Transcript (Figure 7.5).

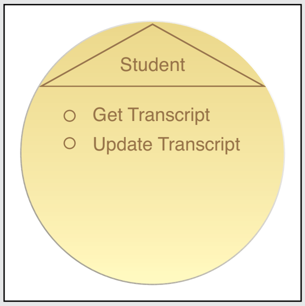

Figure 7.5
The Student service candidate with two service capability candidates.

As previously mentioned, the following three remaining actions are put aside for when utility services are modeled, later in this process:

- Send Rejection Notice
- Send Acceptance Notice
- Print Hard Copy of Award Conferral Record

Step 4: Identify Process-Specific Logic

Process-specific logic is separated into its own logical service layer. For a given business process, this type of logic is commonly grouped into a task service or a service consumer acting as the composition controller.

CASE STUDY EXAMPLE

The following actions are considered non-agnostic because they are specific to the Student Award Conferral business process:

- Initiate Conferral Application
- **Verify Event Details**
- **If Event is Invalid or Ineligible for Award, End Process**
- **Verify Student Transcript Qualifies for Award Based on Award Conferral Rules**
- **If Student Transcript Does Not Qualify, Initiate Rejection**

The first action on this list forms the basis of a service capability candidate, as explained shortly in the Confer Student Award task service candidate description. The remaining actions in bold do not correspond to service capability candidates. Instead, they are identified as logic that occurs internally within the Confer Student Award task service.

Confer Student Award Service Candidate

The Initiate Conferral Application action is translated into a simple Start service capability candidate as part of a Confer Student Award task service candidate (Figure 7.6). It is expected that the Start capability will be invoked by a separate software program, which would be acting as a composition initiator.

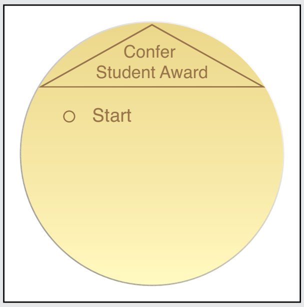

Figure 7.6
The Confer Student Award task service candidate with a single service capability that launches the automation of the Student Award Conferral business process.

Step 5: Identify Resources

By examining the functional contexts associated with individual actions, we can begin to make a list of how these contexts relate to or form the basis of resources. It can be helpful to further qualify identified resources as agnostic (multipurpose) or non-agnostic (single-purpose), depending on how specific we determine their usage and existence are to the parent business process.

Step 3 explained how labeling a service candidate or a service capability candidate as "agnostic" has significant implications as to how we approach the modeling of that service. This is not the case with resources. From a modeling perspective, agnostic resources can be incorporated into agnostic service and capability candidates without limitation. The benefit to identifying agnostic resources is to earmark them as parts of the enterprise that are likely to be shared and reused more frequently than non-agnostic resources. This can help us prepare necessary infrastructure or perhaps even limit their access in how we model (and subsequently design) the service capabilities that encompass them.

Note that resources identified at this stage can be expressed using the forward slash as a delimiter. This is not intended to result in URL-compliant statements; rather, it is a means by which to recognize the parts of service capability candidates that pertain to resources. Similarly, modeled resources are intentionally represented in a simplified form. Later, in the service-oriented design stage, the syntactically correct resource identifier statements are used to represent resources, including any necessary partitioning into multi-part URL statements (as per resource identifier syntax standards being used).

CASE STUDY EXAMPLE

Subsequent to a review of the processing requirements of the service capability candidates defined so far, the following potential resources are identified:

- /Process/
- /Application/
- /Event/
- /Award/
- /Student Transcript/
- **/Notice Sender/**
- **/Printer/**

Before proceeding, the MUA service modeling team decides to further qualify the /Process/ and /Application/ resource candidates to better associate them with the nature of the overarching business processing logic, as follows:

- /Student Award Conferral Process/
- /Conferral Application/

These qualifiers help distinguish similar resources that may exist as other forms of applications or rules.

Because the service modeling process has, so far, already produced a set of entity services, each of which represents a business entity, it is further decided to establish some preliminary mapping between identified resources and entities, as shown in Table 7.1.

Entity	Resource
Event	/Event/
Award	/Award/
Student	/Student Transcript/

Table 7.1
Mapping business entities to resources.

The bolded resources in the preceding list are put aside for when utility services will be modeled, later in this process. Additional resources are not mapped because they do not currently relate to known business entities. They may end up being mapped during future iterations of the service modeling process.

Step 6: Associate Service Capabilities with Resources and Methods

We now associate the service capability candidates defined in Steps 3 and 4 with the resources defined in Step 5, as well as with available uniform contract methods that may have been established. If we discover that a given service capability candidate requires a method that does not yet exist in the uniform contract definition, the method can be proposed as input for the next iteration of the Model Uniform Contract task that is part of the service inventory analysis cycle.

We continue to use the same service candidate and service capability candidate notation, but we append service capability candidates with their associated method plus resource combinations. This allows for a descriptive and flexible expression of a preliminary service contract that can be further changed and refined during subsequent iterations of the service-oriented analysis process.

NOTE

At this stage it is common to associate actions with regular HTTP methods, as defined via uniform contract modeling efforts. Complex methods can be comprised of pre-defined sets and/or sequences of regular method invocations. If complex methods are defined at the service modeling stage, then they can also be associated as appropriate.

CASE STUDY EXAMPLE

The MUA service modeling team continues to expand upon their original service candidate definitions by adding the appropriate uniform contract methods and resources, as follows.

Confer Student Award Service Candidate (Task)

The business document required as the primary input to kick off the Student Award Conferral business process is the application submitted by the student. It was initially assumed that an /Application/ resource would be required to represent this document. However, upon further analysis, it turns out that all the Start service capability candidate needs is a POST method to forward the application document to a resource named after the business process itself (Figure 7.7).

Event Service Candidate (Entity)

The sole Get Details service capability candidate is appended with the GET method and the /Event/ resource (Figure 7.8).

Award Service Candidate (Entity)

The Get Details service capability is correspondingly associated with a GET method plus /Award/ resource combination. The Confer and Update History service capability candidates each require input data that will update resource data, and therefore are expanded with a preliminary POST method and the /Awards/ resource (Figure 7.9). This method may later be refined during the service-oriented design phase.

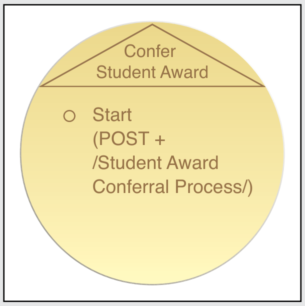

Figure 7.7
The Confer Student Award service candidate with method and resource association.

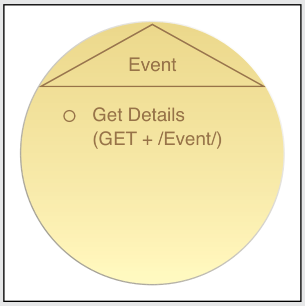

Figure 7.8
The Event service candidate with method and resource association.

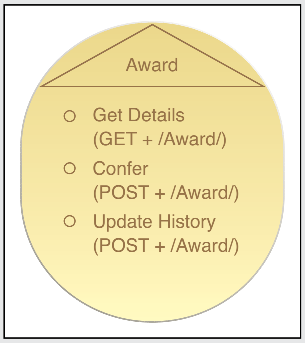

Figure 7.9
The Award service candidate with method and resource associations.

Student Service Candidate (Entity)

The Get Transcript service capability candidate is associated with the GET method and the /Student Transcript/ resource. The Update Transcript is appended with the POST method together again with the /Student Transcript/ resource (Figure 7.10).

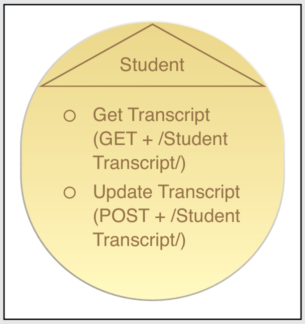
Student
Get Transcript (GET + /Student Transcript/)
Update Transcript (POST + /Student Transcript/)

Figure 7.10
The Student service candidate with method and resource associations.

Step 7: Apply Service-Orientation

The business process documentation we used as input for the service modeling process may provide us with a level of knowledge as to the underlying processing required by each of the identified REST service capability candidates. Based on this knowledge, we may be able to further shape the definition and scope of service capabilities, as well as their parent service candidates, by taking a relevant subset of the service-orientation principles into consideration.

CASE STUDY EXAMPLE

When applying this step, the MUA service modeling team is faced with various practical concerns, based on what participating SOA architects can provide in terms of knowledge of the implementation environment that the services will be deployed in.

For example, they identify that a given set of resources is related to data provided by a large legacy system. This impacts functional service boundaries by the extent to which the Service Autonomy (297) principle can be applied.

Step 8: Identify Service Composition Candidates

Here we document the most common service capability interactions that can take place during the execution of the business process logic. Different interactions are mapped out based on the success and failure scenarios that can occur during the possible action sequences within the business process workflow.

Mapping these interaction scenarios to the required service capability candidates enables us to model candidate service compositions. It is through this type of view that we can get a preview of the size and complexity of potential service compositions that result from how we defined the scope and granularity of agnostic and non-agnostic service candidates (and capability candidates) so far. For example, if we determine that the service composition will need to involve too many service capability invocations, we still have an opportunity to revisit our service candidates.

It is also at this stage that we begin to take a closer look at data exchange requirements (because for services to compose each other, they must exchange data). This may provide us with enough information to begin identifying required media types based on what has already been defined for the uniform contract. Alternatively, we may determine the need for new media types that have not yet been modeled. In the latter case, we may be gathering information that will act as input for the Model Uniform Contract task that is part of the service inventory analysis cycle (as explained later in the *Uniform Contract Modeling and REST Service Inventory Modeling* section).

NOTE

The depth of service compositions can particularly impact method definition. It is important to pose questions about the possible failure scenarios that can occur during service composition execution.

CASE STUDY EXAMPLE

The MUA service modeling team explores a set of service composition scenarios that correspond to success and failure conditions that may arise when the Student Award Conferral process is executed.

Figure 7.11 illustrates the composition hierarchy of service candidates that is relatively consistent across these scenarios. In each case, the Confer Student Award task service invokes the Event, Award, and Student entity services. The Award entity service further composes the Notification utility service to issue acceptance or rejection notifications and, if the award is conferred, the Document utility service to print the award record.

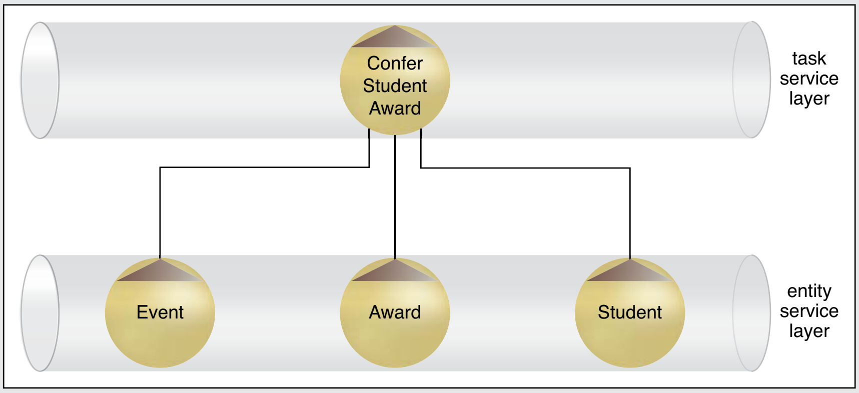

Figure 7.11
A look at the service composition candidate hierarchy that is formed as various service interaction scenarios are explored during this stage.

NOTE

This next series of steps is optional and more suited for complex business processes and larger service inventory architectures. It requires that we more closely study the underlying processing requirements of all service capability candidates in order to abstract further utility service candidates.

Step 9: Analyze Processing Requirements

As mentioned in the description for Step 3, the emphasis so far in this service modeling process will likely have been on business-centric processing logic. This is to be expected when working with business process definitions that are primarily based on a business view of automation. However, it is prudent to look under the hood of the business logic defined so far in order to identify the need for any further application logic.

To accomplish this, we need to consider the following:

- Which of the resources identified so far can be considered utility-centric?
- Can actions performed on business-centric resources be considered utility-centric (such as reporting actions)?
- What underlying application logic needs to be executed in order to process the actions and/or resources encompassed by a service capability candidate?
- Does any required application logic already exist?
- Does any required application logic span application boundaries? (In other words, is more than one system required to complete the action?)

Note that information gathered during the Identify Automation Systems step of the parent service-oriented analysis process will be referenced at this point.

CASE STUDY EXAMPLE

The MUA team carefully studies the processing requirements of the logic that will need to be encapsulated by the service candidates defined so far. They confirm that, beyond the already-identified Send Rejection Notice, Send Acceptance Notice, and Print Hard Copy of Award Conferral Record actions, there appear to be no further utility-centric functions required. This then sets the stage for the upcoming Define Utility Services (and Associate Resources and Methods) step during which these actions, together with the previously identified utility-centric resources, will act as the primary input for utility service candidate definition.

However, while no new utility-centric processing requirements were identified, a concern was raised specifically regarding the non-agnostic Verify Student Transcript Qualifies for Award Based on Award Conferral Rules action that is currently encapsulated as part of the Confer Student Award task service. Architects discover that to complete this action, an external Rules utility service will need to be composed and invoked to complete the verification. Infrastructure statistics show that this existing Rules service is widely used and frequently reaches its usage thresholds, resulting in response delays and, during peak usage periods, occasional response rejections.

This raises concerns by business analysts who point out that there are policy-driven requirements that need to be fulfilled by carrying out an immediate verification of student transcripts. Further, and more importantly, after a verification has occurred, it is legally binding and cannot be reversed.

As a result, the MUA team classifies the Verify Student Transcript Qualifies for Award Based on Award Conferral Rules action as having critical and specialized processing requirements that cannot be met if it were to remain as part of the task service implementation. They therefore determine that this logic needs to be moved to a dedicated microservice.

Step 10: Define Utility Service Candidates (and Associate Resources and Methods)

In this step we group utility-centric processing steps according to pre-defined contexts. With utility service candidates, the primary context is a logical relationship between capability candidates. This relationship can be based on any number of factors, including:

- Association with a specific legacy system
- Association with one or more solution components
- Logical grouping according to type of function

Various other issues are considered after service candidates are subjected to the service-oriented design process. For now, this grouping establishes a preliminary utility service layer in which utility service candidate capabilities are further associated with resources and methods. A primary input will be any utility-centric resources previously defined in Step 5.

NOTE

Modeling utility service candidates is notoriously more difficult than entity service candidates. Unlike entity services where we base functional contexts and boundaries upon already-documented enterprise business models and specifications (such as taxonomies, ontologies, entity relationships, etc.), there are usually no such models for application logic. Therefore, it is common for the functional scope and context of utility service candidates to be continually revised during iterations of the service inventory analysis cycle.

CASE STUDY EXAMPLE

The MUA team proceeds by digging up notes from prior process steps regarding utility-centric actions that have been documented so far. Combined with the research they collected from the Analyze Processing Requirements step, they proceed to define the following two utility services.

Notification Service Candidate

The Send Rejection Notice and Send Acceptance Notice actions are combined into one generic Send service capability candidate as part of a utility service called Notification (Figure 7.12). The Send capability will accept a range of input values, enabling it to issue approval and rejection notifications, among others.

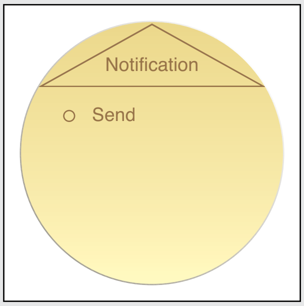

Figure 7.12
The Notification service candidate, with a sole service capability candidate that processes two of the actions identified for the parent business process.

Document Service Candidate

The MUA service modeling team originally created a Document Printing utility service, but then realized its functional scope was too limiting. Instead, it broadened its scope to encompass generic document processing functions. For the time being, this service candidate will only include a Print service capability candidate to accommodate the Print Hard Copy of Award Conferral Record action (Figure 7.13). In the future, this utility service will include other service capabilities that perform generic document processing tasks, such as faxing, routing, and parsing.

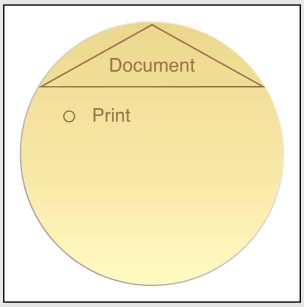

Figure 7.13
The Document service candidate with a generic Print service capability candidate.

Next, the /Notice Sender/ and /Printer/ resources identified earlier in Step 5 are revisited so that they, together with the appropriate methods, can be allocated to the newly defined utility service candidate capabilities.

Notification Service Candidate

The Send service capability candidate is expanded with the POST method and the /Notice Sender/ resource (Figure 7.14).

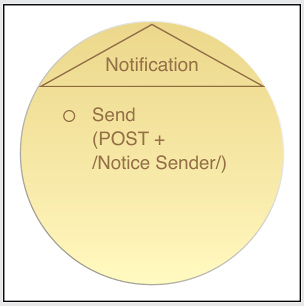

Figure 7.14
The Notification service candidate with method and resource association.

Document Service Candidate

The highly generic Print service capability candidate is expanded with a POST method and the /Printer/ resource (Figure 7.15). Any document sent to the Print capability will be posted to the /Printer/ resource and then printed.

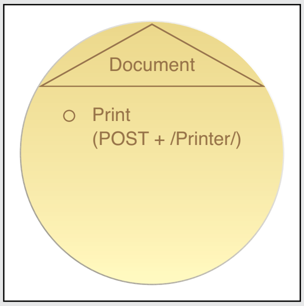

Figure 7.15
The Document service candidate with method and resource association.

Step 11: Define Microservice Candidates (and Associate Resources and Methods)

We now turn our attention to the previously identified non-agnostic processing logic to determine whether any unit of this logic may qualify for encapsulation by a separate microservice. As discussed in Chapter 5, the microservice model can introduce a highly independent and autonomous service implementation architecture that can be suitable for units of logic with particular processing demands.

Typical considerations can include:

- Increased autonomy requirements
- Specific runtime performance requirements
- Specific runtime reliability or failover requirements
- Specific service versioning and deployment requirements

CASE STUDY EXAMPLE

In support of isolating the processing for the Verify Student Transcript Qualifies for Award Based on Award Conferral Rules action, the MUA team establishes a microservice candidate called Verify Application, with a single Verify service capability candidate (Figure 7.16).

Verify Application Service

It is presumed that the eventual implementation environment for this service will be highly autonomous and may include a redundant implementation of the Rules service to guarantee the previously identified reliability requirements.

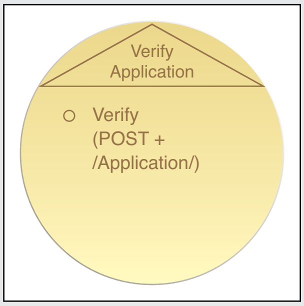

Figure 7.16
The Verify Application service candidate with method and resource association.

Step 12: Apply Service-Orientation

This step is a repeat of Step 7 provided here specifically for any new utility service candidates that may have emerged from the completion of Steps 9 and 10.

Step 13: Revise Candidate Service Compositions

Now we revisit the original service composition candidate scenarios we identified in Step 8 to incorporate new or revised utility service candidates. The result is typically an expansion of the service composition scope where more utility service capabilities find themselves participating in the business process automation.

CASE STUDY EXAMPLE

The Confer Student Award service composition expands with the introduction of the Notification and Document utility services and the Verify Application microservice (Figure 7.17).

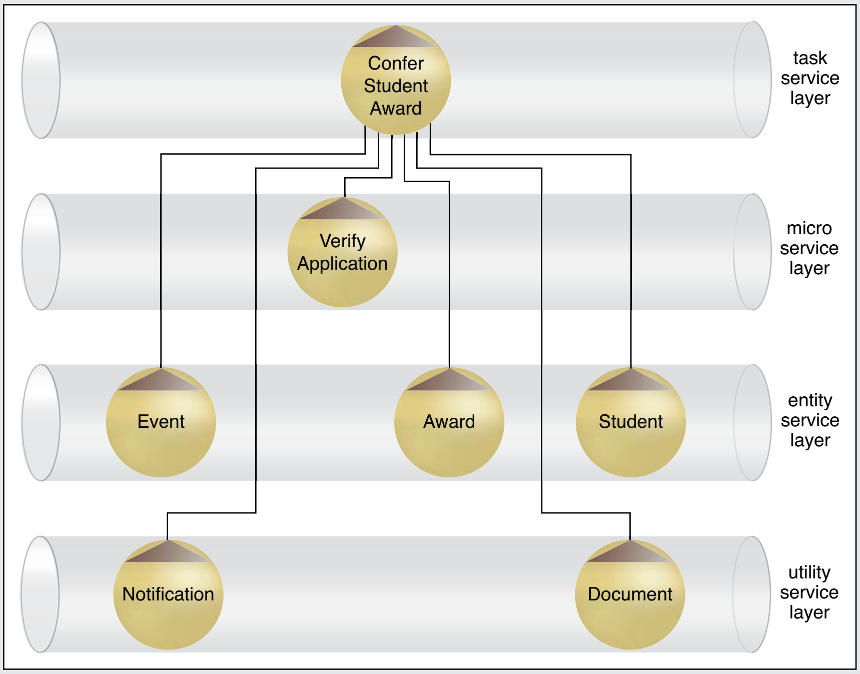

Figure 7.17
The revised service composition candidate incorporating new utility services and a microservice.

Step 14: Revise Resource Definitions and Capability Candidate Grouping

Both business-centric and utility-centric resources can be accessed or processed by utility services and microservices. Therefore, any new processing logic identified in the preceding steps can result in opportunities to further add to and/or revise the set of resources modeled so far.

Furthermore, with the introduction of new utility services and/or microservices, we need to check the grouping of all modeled service capability candidates because:

- Utility service capability candidates defined in Steps 9 and 10 may remove some of the required actions that comprised entity service capability candidates defined earlier, in Step 3.
- The introduction of new utility service candidates may affect (or assimilate) the functional scopes of already-defined utility service candidates.
- The modeling of larger and potentially more complex service composition candidates in Step 13 may lead to the need to reduce or increase the granularity of some service capability candidates.

NOTE

As a result, subsequent execution of several of the modeling steps will require an extra discovery task during which we determine what relevant service candidates, resources, and uniform contract properties exist, prior to defining or proposing new ones.

7.2 Additional Considerations

Uniform Contract Modeling and REST Service Inventory Modeling

A service inventory is a collection of services that are independently owned, governed, and standardized. When we apply the Uniform Contract {311} constraint during an SOA project, we typically do so for a specific service inventory. This is because a uniform contract will end up standardizing a number of aspects pertaining to service capability representation, data representation, message exchange, and message processing. The definition of a uniform contract is ideally performed prior to individual REST service contract design, because each REST service contract will be required to form dependencies on and operate within the scope of the features offered by its associated uniform contract.

Organizations that aim to build a single inventory of REST services will typically rely on a single over-arching uniform contract to establish baseline communication standards. Those that proceed with a domain-based service inventory approach instead will most likely need to define a separate uniform contract for each domain service inventory. Because domain service inventories tend to vary in terms of standardization and governance, separate uniform contracts can be created to accommodate these individual requirements. This is why uniform contract modeling can be part of the service inventory analysis project stage.

The purpose of the service inventory analysis stage is to enable a project team to first define the scope of a service inventory via the authoring of a service inventory blueprint. This specification is populated by the repeated execution of the service inventory analysis cycle. Once all iterations (or as many as are allowed) are completed, we have a set of service candidates that have been (hopefully) well-defined, both individually and in relation to each other. The subsequent step is to proceed with the design of the respective service contracts.

When we know in advance that we will be delivering these services using REST, it is beneficial to incorporate the modeling of the inventory's uniform contract into the modeling of the service inventory itself. This is because as we perform each service-oriented analysis process and model and refine each service candidate and service capability candidate, we gather more and more intelligence about the business automation requirements that are distinct to that service inventory. Some of this information will be relevant to how we define the methods and media types of the uniform contract.

Examples of useful areas of intelligence include:

- Understanding the types of information and documents that will need to be exchanged and processed can help define necessary media types.
- Understanding the service models (entity, utility, task, etc.) in use by service candidates can help determine which available methods should be supported.
- Understanding policies and rules that are required to regulate certain types of interaction can help determine when certain methods should not be used, or help define special features that may be required by some methods.
- Understanding how service capability candidates may need to be composed can help determine suitable methods.
- Understanding certain quality-of-service requirements (especially in relation to reliability, security, transactions, etc.) can help determine the need to support special features of methods, and may further help identify the need to issue a set of pre-defined messages that can be standardized as complex methods.

A practical means of incorporating the task of uniform contract modeling as part of the service inventory analysis is to group it with the Define Technology Architecture step (Figure 7.18). During this step general service inventory architecture characteristics and requirements are identified from the same types of intelligence we collect for the definition of uniform contract features. In this context, the uniform contract is essentially being defined as an extension to the standardized technology architecture for the service inventory.

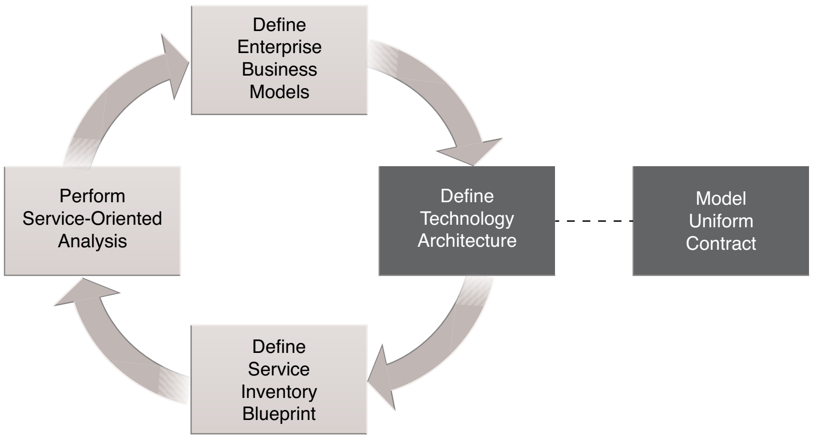
Define Enterprise Business Models
Define Technology Architecture
Model Uniform Contract
Define Service Inventory Blueprint
Perform Service-Oriented Analysis

Figure 7.18
In the service inventory analysis cycle, uniform contract modeling can be included as an iterative task.

If combining the Model Uniform Contract task with the Define Technology Architecture step turns out to be an unsuitable grouping, then the Model Uniform Contract task can be positioned as its own step within the cycle.

When we begin working on the uniform contract definition, one of the key decisions will be to determine the sources to be used to populate its methods and media types. As a general starting point, we can look to the HTTP specification for an initial set of methods and the IANA Media Type Registry for the initial media types. Further media types and possibly further methods may come from a variety of internal and external sources.

NOTE

It is also worth noting that methods and media types can be standardized independently of a service inventory. For example, HTTP methods are defined by the IETF. A service inventory that uses these methods will include a reference to the IETF specification as part of the service inventory uniform contract definition. Media types may be specified on an ongoing basis by external bodies, such as the W3C, the IETF, industry bodies across various supply chains, or even within an IT enterprise.

Note that the asterisk symbol can be used in the top-right corner to indicate that a REST service candidate is being modeled during this step that either:

- Incorporates methods and/or media types already modeled for the uniform contract, or
- Introduces the need to add or augment methods and/or media types for the uniform contract

This type of two-way relationship between the Perform Service-Oriented Analysis step (which encompasses the REST service modeling process) and the Model Uniform Contract task is a natural dynamic of the service inventory analysis cycle.

NOTE

It is usually during the Model Uniform Contract task that a uniform contract profile is first populated with preliminary characteristics and properties. This profile document is then further refined as the uniform contract and is physically designed and maintained over time.

REST Constraints and Uniform Contract Modeling

Although REST constraints are primarily applied during the physical design of service architectures, taking them into consideration as the uniform contract takes shape during the service-oriented analysis stage can be helpful. For example:

- *Stateless {308}* – From the data exchange requirements we are able to model between service candidates, can we determine whether services will be able to remain stateless between requests?
- *Cache {310}* – Are we able to identify any request messages with responses that can be cached and returned for subsequent requests instead of needing to be processed redundantly?

- *Uniform Contract {311}* – Can all methods and media types we are associating with the uniform contract during this stage be genuinely reused by service candidates?
- *Layered System {313}* – Do we know enough about the underlying technology architecture to determine whether services and their consumers can tell the difference between communicating directly or communicating via intermediary middleware?

The extent to which concrete aspects of REST constraint application can be factored into how we model the uniform contract will depend directly on:

- The extent to which the service inventory technology architecture is defined during iterations of the service inventory analysis cycle, and
- The extent to which we learn about a given business process's underlying automation requirements during Step 2 of the service-oriented analysis process

Much of this will be dependent on the amount of information we have and are able to gather about the underlying infrastructure and overall ecosystem in which the inventory of services will reside. For example, if we know in advance that we are delivering a set of services within an environment riddled with existing legacy systems and middleware, we will be able to gain access to many information sources that will help determine boundaries, limitations, and options when it comes to service and uniform contract definition. On the other hand, if we are planning to build a brand-new environment for our service inventory, there will usually be many more options for creating and tuning the technology architecture in support of how the services (and the uniform contract) can best fulfill business automation requirements.

SOA PATTERNS

When determining the scope of a service inventory and whether multiple service inventories are allowed within an enterprise environment, the decision usually comes down to whether the Enterprise Inventory [340] or the Domain Inventory [338] pattern is applied.

REST Service Capability Granularity

When actions are defined at this stage, they are considered fine-grained in that each action is clearly distinguished with a specific purpose. However, within the scope of that purpose they can often still be somewhat vague and can easily encompass a range of possible variations.

Defining conceptual service candidates using this level of action granularity is common with mainstream service modeling approaches. It has proven sufficient for SOAP-based Web services because service capabilities that need to support variations of functionality can still be effectively mapped to WSDL-based operations capable of handling a range of input and output parameters.

With REST service contracts, service capabilities are required to incorporate methods (and media types) defined by an overarching uniform contract. As already discussed in the preceding section, the uniform contract for a given service inventory can be modeled alongside and in collaboration with the modeling of service candidates, as long as we know in advance that REST will act as the primary service implementation medium.

Whereas a WSDL-based service contract can incorporate custom parameter lists and other service-specific features, REST puts an upper bound on the granularity of message exchanges at the level of the most complex or most general purpose method and media type. This may, in some cases, lead to the need to define finer-grained service capabilities.

Figure 7.19 highlights the difference between a service candidate modeled in an implementation-neutral manner versus one modeled specifically for the REST service implementation medium.

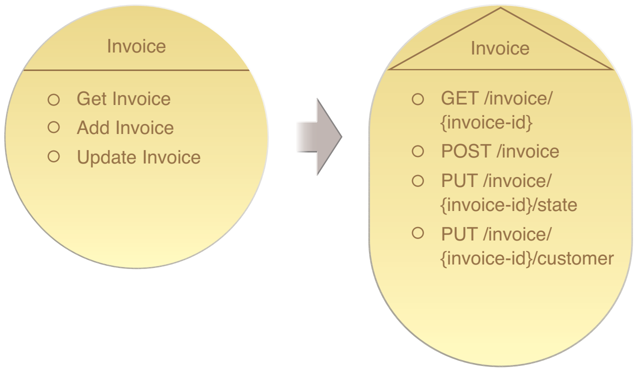
Invoice
Get Invoice
Add Invoice
Update Invoice
Invoice
GET /invoice/{invoice-id}
POST /invoice
PUT /invoice/{invoice-id}/state
PUT /invoice/{invoice-id}/customer

Figure 7.19

A REST service candidate can be modeled specifically to incorporate uniform contract characteristics. The Update Invoice service capability candidate is split into two variations of the PUT /invoice/ service capability: one that updates the invoice state value, and another that updates the invoice customer value.

Resources vs. Entities

Part of the REST service modeling process explores the identification of resource candidates. It is through the definition of these resource candidates that we begin to introduce a Web-centric view of a service inventory. Resources represent the "things" that need to be accessed and processed by service consumers.

What we are also interested in establishing during the service-oriented analysis stage is the encapsulation of entity logic. As with resources, entities also often represent "things" that need to be accessed and processed by service consumers.

What then is the difference between a resource and an entity? To understand REST service modeling, we need to clearly understand this distinction:

- Entities are business-centric and are derived from enterprise business models, such as entity relationship diagrams, logical data models, and ontologies.
- Resources can be business-centric or non-business-centric. A resource is any given "thing" associated with the business automation logic enabled by the service inventory.
- Entities are commonly limited to business artifacts and documents, such as invoices, claims, customers, etc.
- Some entities are more coarse-grained than others. Some entities can encapsulate others. For example, an invoice entity may encapsulate an invoice detail entity.
- Resources can also vary in granularity, but are often fine-grained. It is less common to have formally defined coarse-grained resources that encapsulate fine-grained resources.
- All entities can relate to or be based on resources. Not all resources can be associated with entities because some resources are non-business-centric.

The extent to which we need to formalize the mapping between business-centric resources and entities is up to us. The REST service modeling process provides steps that encourage us to define and standardize resources as part of the service inventory blueprint so that we gain a better understanding of how and where resources need to be consumed.

From a pure modeling perspective we are further encouraged to relate business-centric resources to business entities so that we maintain a constant alignment with how business-centric artifacts and documents exist within our business. This perspective is especially valuable as the business and its automation requirements continue to evolve over time.

This page intentionally left blank